**LAB- Understanding Namespaces**

# In this lab you will be understanding how to use namespaces

Before proceeding, make sure u create a ubuntu VM .

I will be using GCP for creating a VM

**STEP 1 – Running process in pid namespace**

Ssh into ubuntu vm

**Create p1.sh and p2.sh using vi editor using the contents at using below links:**

[**dockertraining/p1.sh**](https://raw.githubusercontent.com/sivaprasadvalluru/docker/master/p1.sh)

[**dockertraining/p2.sh**](https://raw.githubusercontent.com/sivaprasadvalluru/docker/master/p1.sh)

run process p1 in background by executing below command

**./p1.sh &**

Run process p2 also in background by executing below command:

**./p2.sh**

You should observe that a file with name p2out.txt created

Execute the below command to see the contents of it

**Cat p2out.txt**

You should observe that pid of p1 is written.

Now execute **ps –af** and onserve that pid of p1 is same as the one written in p2out.txt

This means that process p2 is able to see process p1. This is because both the processes are executed in same default pid namespace

Execute the below command to see the pid namespace id

**lsns –t pid**

We want to create run p1 and p2 processes in their own isolated namespaces

ssh in to your ubuntu machine from two terminals

In first terminal, we want to run p1 in its own pid namespace

Execute the below command to start a new pid namespance

**sudo unshare –fp --mount-proc bash**

Now execute **lsns –t pid** and observe that the namespace ids are different

Execute p1 process using below command:

./p1.sh &

Observe the pid of process p1 by executing **ps –af** command

From terminal 2, execute ps –af and observe that u see the process for p1.sh is listed.

Observe that the PID shown in this terminal 2 is different than the one isn terminal 1

From the parent pid namespace, we can see all the processes of child pid namespace. But the process ids will be different.

In terminal 2 , execute the below command to start another pid namespace

**sudo unshare –fp --mount-proc bash**

now execute process p2 by executing ./p2.sh &

Now, as we are running p2 in a separate namespace, it cannot see the process p1

Execute **“Cat p2out.txt”** command and observe that below is written to the file

“unable to find pidofp1:

So, we understood how to create pid namespace using unshare command

Type exit to exit the namespace

**STEP 2 – Creating mount namespace and setting root directory using pivot\_root**

Run df –a command and see where /dev/sda1 is mounted

Create directories with name “a” and “b” using below commands.

mkdir a

mkdir b

cd into directory a and create 1.txt and 2.txt inside directory a using touch 1.txt and touch 2.txt

cd into directory b and create 3.txt and 4.txt inside directory b using touch 3.txt and touch 4.txt

We want to mount an existing directory “b” into directory “a” using bind mount.

Use the below command:

**sudo mount –bind b a**

Now ls a/ and observe that u see 3.txt and 4.txt inside a

So, directory b is mounted on to directory a

Execute the below command to unmount directory a

**Sudo umount a**

By default, there will be a default mount namespace

execute **lsns** and observe the mount namespace id

We want to create a new mount namespace along with pid namespace

Execute below command in terminal 1

**Sudo unshared –fpm –mount-proc bash**

Now, execute lsns and observe the mount namespace id

You should observe that this mount namespace id is different than the parent mount namespace

Now , in this new mountspace we want to mount directory a on to b using bind mount.

Execute below command in terminal 1

**sudo mount –bind a b**

Now ls b/ and observe that u see 1.txt and 2.txt inside a

So, directory a is mounted on to directory b

This is inside a different mount namespace.

In Terminal 2, ssh into ubuntu machine .

Now, you will be in the parent namespace.

do execute ls b/ .

You should observe original 3.txt and 4.txt because mount was done in a different namespace earlier and that does not affect the parent namespace

We want to set this directory b as the root directory

Change the directory to “b” using “cd b”

Create a directory with name **old\_root** using **mkdir old\_root**

Now execute the below command to make

**sudo mount –bind a a**

sudo pivot\_root . old\_root

Now try to list the contents of root directory using ls /

You should see the contents of b only

Now, type **exit** to come out of the namespace

**STEP 3 – Enter into existing namespace**

Create a process and mount namespace using below command in terminal 1:

**Sudo unshared –fpm –mount-proc bash**

In terminal 2, we are in parent namespace.

We want to enter into namespace in terminal 1 from terminal 2

From terminal 2 where you are in parent namespace, execute **ps –a** and find the pid of bash

To enter into the namespace of that bash process, and list the contents of / execute below command:

sudo nsenter –a –t <<pidofbash>> ls /

**STEP 4 – overlay filesystem**

Create a folder with name **overlays/alpine** and cd into it.

Download alpine using below command:

wget <https://dl-cdn.alpinelinux.org/alpine/v3.17/releases/x86_64/alpine-minirootfs-3.17.2-x86_64.tar.gz>

extract the file using tar –xf alpine-minirootfs-3.17.2-x86\_64.tar.gz

now delete the tar file

You have all the alpine related files in this folder **alpine**.

Now change into overlays folder

Create 3 directories using below command:

**mkdir lowerdir upperdir mergeddir workdir**

cd lowerdir

create a file with name lower1.txt with text “this is from lower1.txt from lowerdir”

create a file with name lower2.txt with text “this is from lower2.txt from lowerdir”

cd to upperdir

create a file with name upper1.txt with text “this is from upper1.txt from lowerdir”

create a file with name upper.txt with text “this is from upper2.txt from upperdir”

create a folder with name app1 and cd into it

create a file with name app.txt with text “this is from app.txt of app1”

create a folder with name app2 and cd into it

create a file with name app.txt with text “this is from app.txt of app2”

Now We want to mount lowerdir,app1 and upperdir into mergeddir using overlay filesystem.

Cd to overlay directory

Execute the below command:

**sudo mount -t overlay -o lowerdir=lowerdir/:app1/,upperdir=upperdir/,workdir=workdir/ none mergeddir/**

Now execute df-a and observe that a mount with name mergeddir is created.

Execute ls mergeddir and observe that u see all the files from lowerdir,app1,upperdir

Now modify lower1.txt in mergeddir.

Observe that lower1.txt is not changed in lowerdir. But a file with same name is created in upperdir.

Delete the file lower1.txt in mergeddir. Observe that the file is not deleted from lowerdir.

ls -al upperdir. What did u observe in upperdir. Is the deleted file present?

**STEP 5 – Controlling memory usage using CGroup**

Install Cgroup utilities using below command:

**sudo apt-get install cgroup-bin cgroup-lite cgroup-tools cgroupfs-mount libcgroup1**

**ls /sys/fs/cgroup** to view all the cgroups

Now cd into **/sys/fs/cgroup/memory**

We want to create a memory controller with name srinimemlimiter. Execute the below command:

**sudo cgcreate –g memory:srinimemlimiter**

observe that there is a folder with name srinimemlimiter created inside /sys/fs/cgroup/memory

We want to set a memory limit of 700 MB for this controller srinimemlimiter.

Execute below command:

**sudo cgset –r memory.limit\_in\_bytes=700M srinimemlimiter**

In home directory, download a java jar file using below command:

[DockerTraining/MemoryConsumer.jar](https://github.com/sivaprasadvalluru/docker/raw/master/MemoryConsumer.jar)

Install java 11 if not installed

To run this java program inside the controlgroup which we created, execute the following command:

**sudo cgexec -g memory: srinimemlimiter java –Xmx1024m –jar MemoryConsumer.jar**

This program will exit once the memory consumed crosses 700MB + some swap space

Once program exits because of out of memory, u can see oom counts in **/sys/fs/cgroup/memory/srinimemlimiter/memory.oom\_control**

**STEP 6 – Running namespace inside a cgroup**

**sudo cgexec -g memory:srinimemlimiter unshare -fmp --mount-proc**

Once u get the prompt, u can run applications inside this namespace

**STEP 7 – Controlling cpu usage using CGroup**

**sudo cgcreate –g cpu:srinicpulimiter**

**ls /sys/fs/cgroup/cpu/srinicpulimiter**

Observe that there are files with name cpu.cfs\_quota\_us and cpu.cfs\_period\_us

**cat /sys/fs/cgroup/cpu/srinicpulimiter/cpu.cfs\_period\_us** and observe that the value is 100000

**cat /sys/fs/cgroup/cpu/srinicpulimiter/cpu.cfs\_quota\_us** and observe that the value is -1

set the value in cpu.cfs\_quota\_us as 25000 using the below command so that 25% of cpu will be allocated to any processes running under this cpu cgroup

**sudo cgset –r cpu.cfs\_quota\_us=25000 srinicpulimiter**

Now u can execute any process under this cgroup

Download jar file using below link:

[DockerTraining\Cpuloader.jar](https://github.com/sivaprasadvalluru/docker/raw/master/Cpuloader.jar)

Now u can execute this program under the cgroup using below command:

**sudo cgexec –g cpu:srinicpulimiter java –jar Cpuloader.jar**

use the command top to see the cpu usage and observe that only 25% of cpu is used by this java process

**STEP 7 –controlling CPU usage and memory usage in docker**

docker run --name my-container --cpu-period=50000 --cpu-quota=25000 --memory=512m my-image